# MCTS Overview 687

Sean Folan

November 2024

## 1 Overview

Monte Carlo Tree Search (MCTS) is an online algorithm that attempts to estimate the $q$ function of a state though random simulation. Everytime it visits a state, it builds an ExpectiMax search tree incrementally. The search can be terminated after a given amount of time or an amount of expanded nodes. Typically more useful for huge state spaces, famously used in AlphaGo (Go) and Pluribus (No limit Texas Hold'em Poker).

## 2 Algorithm

The Algorithm has four parts, which are repeated until the computational budget is met.

1. **Selection** - Select an unexpanded node.
2. **Expansion** - If we are not in a terminal state, we expand one or more of the children nodes
3. **Simulation** - Choose one of the new nodes and perform Monte Carlo simulation of the MDP
4. **Backpropagation** - The return is backpropagated up to the root

Steps one and two are defined by a TreePolicy which tells the algorithm how to select and expand and step three utilizes DefaultPolicy which encodes how the simulations are carried out.

### 2.1 Upper Confidence Trees

One of the most popular tree policies is Upper Confidence Trees ($UCT$). This strategy has us pick nodes to...

$$\arg\max_{a \in A} Q(s,a) + 2C_p\sqrt{\frac{2\ln N(s)}{N(s,a)}}$$

## 3 Pseudocode

**Algorithm 1** MCTS

```
Input: MDP M = (S, A, p, d_0, R, γ), Time limit T, current state s_0
Output: Estimated Q function
while time < T do
    node ← Select(s_0)              ▷ Find a node that is not fully explored
    child ← Expand(node)            ▷ Expand the node to get the node you will start the Simulation from
    G ← Simulate(child)             ▷ Run the episode getting return G
    Backpropagate( node, G)         ▷ Return results all the way up to the parent node
end while
```

# 4 Really Simple GridWorld

I first tested my MCTS algorithm on a domain called "Really_Simple_GridWorld" which was a gridworld with deterministic transitions. This made it simple to try different hyperparameters and see their effects in a easy to understand deterministic domain.

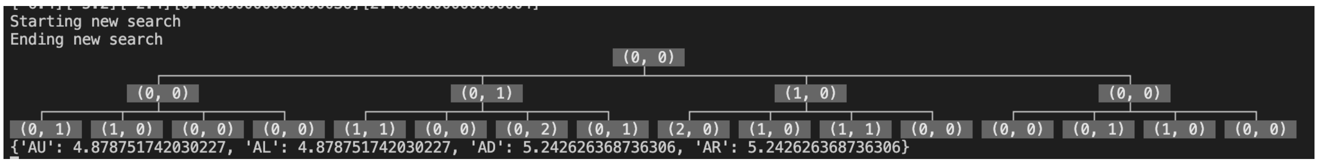


Figure 1: A Tree made MCTS starting at state (0,0) in Really_Simple_GridWorld

# 5 Hyperparameter Search

- Exploration Factor $C_p$
  - Very Little Exploration (0.5)
  - Little Exploration (1)
  - Average Exploration ($\sqrt{2}$)
  - More Exploration (2)
  - Most Exploration (5)
- Default Policy

  Default Policy is the way that that outcomes are simulated in the Simulate step of MCTS. A better Default policy allows for better estimates to be made of a state's reward. I tried three general types of policies Random, Majority AR/AD and State Score which gave each state a score and prioritized going to higher scored states. All three domains are similar so I will summarize the general idea of the policies below.
  - Random
  - Majority AR/AD

    My second idea was that since in general the domains started in the top left and wanted to go to the bottom right, it should in general go down and right.
  - State Score

    The idea that works the best is giving states a score by some function. If a state is closer to the goal state, it gets plus points, closer to a monster state is minus points. I also made it so that transitioning into the same state is less desirable.
- Tree Size To tune the hyperparameter for Tree Size, I did a uniform random search over 1 to 200 tree expansions.
  - Really Simple GridWorld

    There seems to be a slight correlations with Tree Size and the Length of the path being found by MCTS. Most of the time, even with a small tree it manages to find the shortest path but there are more outliers.

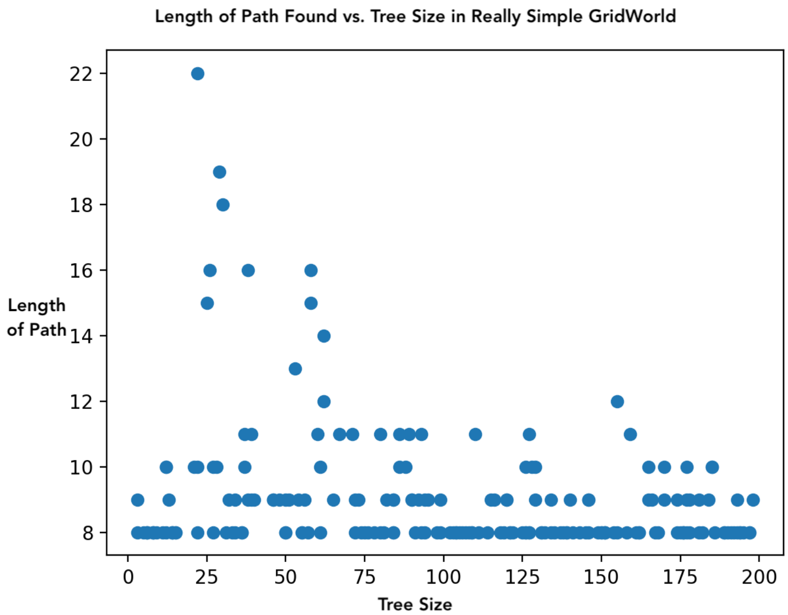


Figure 2:

# 6 Sources


1. https://gibberblot.github.io/rl-notes/single-agent/mcts.html
2. http://incompleteideas.net/book/RLbook2020.pdf
3. http://www.incompleteideas.net/609%20dropbox/other%20readings%20and%20resources/MCTS-survey.pdf
4. https://courses.cs.washington.edu/courses/cse473/11au/slides/cse473au11-adversarial-search.pdf